10/30/2025

MET CS 767 Assignment 2: Neural Nets Intro

Brendan Torok

1. How I modified data and/or code to attempt improvement

1.1 Description of your modifications and reason(s) it could be an improvement

For this assignment I used the base neural network provided in the example code and applied it to my custom dataset as a starting point for training my neural network for my chosen application.

My main changes are as follows:

Increased layers to 4 layers – this increased performance of the neural network although it seems to have made the neural network learn very quickly, within 1-2 epochs. This is an architectural change.

Added manual learning rate tuning to Adam optimizer. This greatly improved the performance of the model. This is a parametric change.

```
optimizer =  tf.keras.optimizers.Adam(learning_rate=0.0004)
```

Changed activation functions to try gelu, and leaky activation. These both decreased performance so I kept the relu activation function. This is an architectural change.

Added L2 regularization and tuned regularization parameter to improve performance. This is both an architectural and parametric change.

```
tf.keras.layers.Dense(128, activation='relu',
                          kernel_regularizer=tf.keras.regularizers.l2(8e-4))
```

Adjusted dropout to 0.10 and 0.05 after each layer, changed from 0.2. This is a parametric change.

```
tf.keras.layers.Dropout(0.10),
    tf.keras.layers.Dense(128, activation='relu',
                          kernel_regularizer=tf.keras.regularizers.l2(8e-4)),
    tf.keras.layers.Dropout(0.05)
```

Added three more layers to the neural network, two with more neurons and one with the same number of neurons in the layer. This was an improvement and is an architectural change.

In training I adjusted the number of epochs and tested early stopping. I discovered that adding more epochs did not greatly improve my performance, but adding early stopping caused the neural network training to stop after 1 or 2 epochs, resulting in poor performance. I settled with 20 epochs as it seemed to have the most stability.

I also made changes to the data by feature engineering to increase signal. This greatly increased the performance of the model, but is not an architectural or parametric change since it is not a change directly to the neural network model itself.

Note: while doing experimenting I did not set random seed. I went back and set a random seed and a few of the results changed. Unfortunately I did not have time to redo all my results with a random seed set.

1.2 Comparison of the result with the original output

Model 1 results:

```
Macro F1: 0.5871
Weighted F1: 0.5958
Additional metrics in Appendix 2
```

Final model results:

```
Macro F1: 0.7270
Weighted F1: 0.7424
Additional metrics in Appendix 2
```

Looking at the results we can see that the macro F1 and weighted F1 scores are greatly increased in the final model. Additionally, the final model also greatly improved the precision of all three classes, particularly the neutral class. However, the final model still struggled to discriminate between the neutral class and the two other classes and could use further tuning and feature engineering to make these features more linearly separable. The stop 5 stocks (see appendix 2) are also different, however SNDK appears in the output of both stocks suggesting that the starting model still can pick up some signal from stocks that are very likely to outperform – at the time of writing SNDK is up 11.89% in the trading day after training this data. The average percentage gain for the top 5 stocks picked is 6.96%, while the benchmark S&P500 has a gain of 1.54%. However, this is obviously a small sample size and likely is simply because the neural network heavily weighs stocks that have high current momentum and are therefore most likely to increase in value in the near-term.

1.3 URL of your Colab code

Code is uploaded into github instead as colab was not used. The code can be run sequentially, although for the comparison the initial data cleaning and test-train split must be run again in order to get the performance of the first model as it was initially trained with less features and fewer performance metrics.

https://github.com/btorok-bu/METCS767_hw2/blob/main/cs767_btorok_hw2.ipynb

2. New Neural Net Application

2.1 Description of the application

This neural network uses a sequential neural network model and aims to predict stock performance (outperform, neutral, underperform) relative to an S&P500 benchmark performance. The data is sourced from yahoo finance and the massive API. The data contains current stock fundamental statistics, and snapshot market activity from 4 years ago, 2 years ago, 1 year ago, 6 months ago, 3 months ago, and 1 month ago. Through feature engineering the performance of each stock relative to the baseline in these time ranges is computed, standardized, weighted, then a momentum score is calculated. The stocks are then labeled based on the percentile rank of their momentum score.

The feature engineering is improved by adding surge ratios which indicate high volume. To improve momentum metrics further, features for the distance to the 52-week high in addition to distance from the 50- and 200-day moving averages were added. Additionally, a metric for the ratio of the float size to the shares outstanding was added since stocks that have a high number of shares outstanding are currently being shorted, indicating that current outlook for the stock may be poor.

After training the model the performance is measured using loss, overall accuracy, macro F1 score, weighted F1 score, classification report containing the precision and recall for all three classes, and confusion matrix. The probability of the overperform class is then used to get the margin over other classes. The mean of the margin and probability to overperform is then taken to get a confidence score. Stocks are then sorted by their confidence score and returned alongside key metrics. With further tuning and additional data this neural network can aid in discovering stocks that have the highest probability to outperform a chosen benchmark index in a given term.

2.2 Summary of your architecture tuning

For my architecture tuning I first built the initial model given in the class example, I then experimented by changing the output layer to use softmax activation and output 3 results for the 3 classes. I then experimented with adding two additional hidden layers. I also experimented with different activation functions and adding a kernel initializer and then adding an L2 kernel regularizer. I also experimented with trying different optimizers like Adam, Nadam, and SGD optimizers. I found the performance was best with the Adam optimizer with a manual learning rate set, RelU activation function, 4 layers, with L2 regularizer.

2.3 Summary of your hyperparameter tuning

For my hyperparameter tuning I tried manual learning rates and found that higher learning rates resulted in better performance for my model with the best learning rate being 0.0005. I also found that the best performance for each layer was 512, 256, 128, and 128. The best dropout value was 0.10, 0.10, 0.10 and 0.05 for the final layer. The best L2 regularizer parameter was 8e-4.

2.4 Three illustrative input/output pairs from running the implementation

1) creating distance from 50 moving average, distance from 200 moving average, and moving average cross features. This feature provides another metric of the direction of the stock movement and whether the moving averages have been crossed and in what direction.

Input:

```
currentPrice  fiftyDayAverage  twoHundredDayAverage

       166.50         157.2214             150.05576
        65.99          71.1404              68.28100
       123.62         131.4586             130.89246
       218.04         222.3706             199.54900
       250.10         246.7902             296.32050
```

Output:

```
currentPrice  fiftyDayAverage  twoHundredDayAverage  dist_ma50  dist_ma200  ma_cross
       166.50         157.2214             150.05576   0.059016    0.109588  0.047753
        65.99          71.1404              68.28100  -0.072398   -0.033553  0.041877
       123.62         131.4586             130.89246  -0.059628   -0.055561  0.004325
       218.04         222.3706             199.54900  -0.019475    0.092664  0.114366
       250.10         246.7902             296.32050   0.013411   -0.155981 -0.167151
```

2) Use the model results to get the probability of being in the overperform class, then get the margin that the p_over is greater than the greatest of the next class. The mean of these two is used to get the confidence. This way high confidence is only given for stocks with high p_over and high margin over next.

Input:

```
     symbol    p_over  margin_over_vs_next
604    CYTK  0.998080             0.996282
345    NVDA  0.966754             0.960818
1397   SNDK  0.915854             0.904881
398    HOOD  0.905623             0.890312
643     FIX  0.903497             0.886419
```

Output:

```
    symbol  confidence
604    CYTK    0.997181
345    NVDA    0.963786
1397   SNDK    0.910367
398    HOOD    0.897967
643     FIX    0.894958
```

3) Input is the performance metrics from each epoch, the output is the learning curve, using just accuracy, loss, and val_accuracy

Input: metrics from epoch in appendix 3

Output: output is figure in appendix 3

2.5 Key code snippets, with explanation

```
model_25 = tf.keras.models.Sequential([
```

```
    # added l2 regularizer with manual tuning of reg param
    tf.keras.layers.Flatten(input_shape=(X_train_proc.shape[1],)),
    tf.keras.layers.Dense(512, activation='relu',
                          kernel_regularizer=tf.keras.regularizers.l2(8e-4)),
    tf.keras.layers.Dropout(0.10),
    tf.keras.layers.Dense(256, activation='relu',
                          kernel_regularizer=tf.keras.regularizers.l2(8e-4)),
    tf.keras.layers.Dropout(0.10),
    tf.keras.layers.Dense(128, activation='relu',
                          kernel_regularizer=tf.keras.regularizers.l2(8e-4)),
    tf.keras.layers.Dropout(0.10),
    tf.keras.layers.Dense(128, activation='relu',
                          kernel_regularizer=tf.keras.regularizers.l2(8e-4)),
    tf.keras.layers.Dropout(0.05)
])

# put together NN with training process, loss, and means of evaluation
# use loss of sparse categorical crossentropy since Dense is in layer
# adjust learning rate in optimizer manually to have slower learning rate
optimizer =  tf.keras.optimizers.Adam(learning_rate=0.0001)
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model_25.compile(optimizer=optimizer,
                loss=loss_fn,
                metrics=[metric.SparseCategoricalAccuracy(name='acc'),
                         metric.SparseTopKCategoricalAccuracy(k=2, name='top2_acc'),
                         # per class percision and recall
                         metric.Precision(name='prec_under', class_id=0),
                         metric.Recall(name='rec_under', class_id=0),
                         metric.Precision(name='prec_neutral', class_id=1),
                         metric.Recall(name='rec_neutral', class_id=1),
                         metric.Precision(name='prec_over', class_id=2),
                         metric.Recall(name='rec_over', class_id=2)
                         ])
predictions = model_25.predict(X_train_proc[:1])
tf.nn.softmax(predictions)

model_25.fit(X_train_proc, y_train, epochs=20, verbose=0)
```

The major adjustments here are to change the dropout parameters, add additional layers, add regularization, add additional metrics for each class, and adjust the learning rate for the Adam optimizer. The epochs were also increased but this did not make much of a difference.

Note: A strong starting AI prompt included in appendix 4 was used to consistently excellent AI responses that ensured the code produced did not use different neural network models to get better performance with this dataset.

2.6 URL of your code

https://github.com/btorok-bu/METCS767_hw2/tree/main

Evaluation

Note: We focus the **quality** (not quantity) of your submission. Material that doesn't respond to the criteria reduces grades.

Criterion *(based on the value of your significant prompts and your text)*	***D***	***C***	***B***	***A***	**Letter Grade**	**%**
Extent to which you ***added technical content is functional and accurate, and shows demonstrably that you understand the AI output, and have learned technically.***	***Low*** *extent.*	***Satisfatory*** *extent*	***Good*** *extent*	*Went significantly* ***beyond*** *what's required.*		*0.0*
Extent to which ***every significant claim you made is supported with clear, relevant logical reasoning that explains its validity.***	***Low*** *extent.*	***Satisfatory*** *extent*	***Good*** *extent*	*Went significantly* ***beyond*** *what's required.*		*0.0*
Extent to which ***your added material probes the key mechanisms in depth.***	***Low*** *extent.*	***Satisfatory*** *extent*	***Good*** *extent*	*Went significantly* ***beyond*** *what's required.*		*0.0*
					Assignment Grade:	*0.0*

The resulting grade is the average of these, using A+=100 (outstanding--rare), A=95 (excellent in all ways), A-=90 (excellent), B+=87 (excellent / very good), B=85 (very good), B-=80 (good) etc.

Appendix 1

Link to AI chat for getting data: https://chatgpt.com/share/690fb2bf-acf8-8006-ab93-267e4f24c916

Link to AI chat for neural network training: https://chatgpt.com/share/6912ca0e-d138-8006-9edb-6df95f567094

Note: a separate AI rules document was used for both chats. The AI rules document was refered throughout the conversation by the LLM.

Appendix 2

First model additional metrics:

```
              precision    recall  f1-score   support

       under       0.67      0.62      0.64       104
     neutral       0.39      0.63      0.48        83
        over       0.81      0.52      0.63       113

    accuracy                           0.58       300
   macro avg       0.63      0.59      0.59       300
weighted avg       0.65      0.58      0.60       300

Confusion matrix:
[[64 36  4]

 [21 52 10]
 [10 44 59]]
```

```
Top 5 stocks:
     symbol    p_over  confidence recommendationKey  currentPrice   forwardPE
604    CYTK  0.982289    0.977802               buy         63.59  -11.797774
345    NVDA  0.935958    0.927752        strong_buy        202.49   49.148060
643     FIX  0.903085    0.892260        strong_buy        965.58   56.832256
1397   SNDK  0.880954    0.869852               buy        199.33   19.709223
51      AZO  0.881337    0.846761               buy       3674.43   21.078648
```

Final model additional metrics:

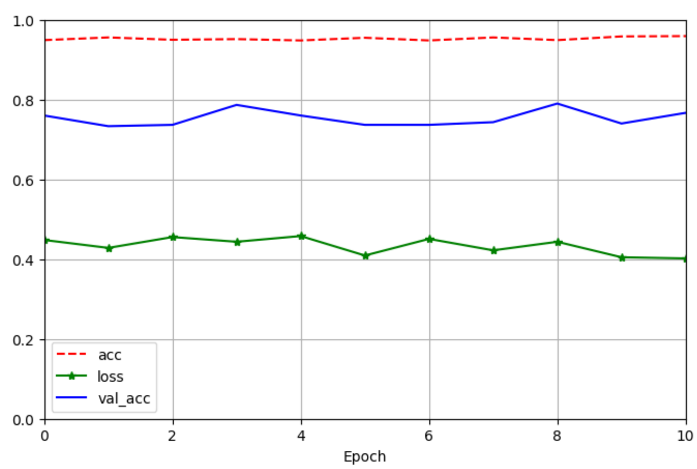
```
              precision    recall  f1-score   support

       under       0.79      0.76      0.77       104
     neutral       0.54      0.58      0.56        83
        over       0.86      0.84      0.85       113

    accuracy                           0.74       300
   macro avg       0.73      0.73      0.73       300
weighted avg       0.75      0.74      0.74       300

Confusion matrix:

[[79 24  1]
 [20 48 15]
 [ 1 17 95]]

Top 5 stocks:

398    HOOD  1.000000    1.000000               buy        146.78  201.068480
604    CYTK  1.000000    1.000000               buy         63.59  -11.797774
1397   SNDK  1.000000    1.000000               buy        199.33   19.709223
358    PLTR  1.000000    1.000000              hold        200.47  426.531920
314      MU  1.000000    1.000000               buy        223.77   17.386948
```

Appendix 3

Input epoch metrics (only one epoch shown):

```
Epoch 1/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - acc: 0.9492 - loss: 0.4347 - prec_neutral: 0.7151 - prec_over: 0.6916 - prec_under: 0.5267 - rec_neutral: 0.7088 - rec_over: 0.5465 - rec_under: 0.4736 - top2_acc: 0.9841 - val_acc: 0.7767 - val_loss: 1.1724 - val_prec_neutral: 0.5417 - val_prec_over: 0.6524 - val_prec_under: 0.8453 - val_rec_neutral: 0.7312 - val_rec_over: 0.5245 - val_rec_under: 0.5977 - val_top2_acc: 0.9800
```

Output graph:



The above graph shows that the performance of the neural network does not improve after the first 1-2 epochs. The model converged very fast so improvements can only be made through more feature engineering or further tuning and adjusting neural network architecture.

Appendix 4

Initial prompt:

Given the ai_rules.txt document included and this code I have currently, how should I build my label column? I would like to label stocks as outperform, neutral, or underperform using data from the given time ranges for each stock. I unfortunately do not have all fundamental data for each time range, and only have the price history and volume for the time ranges and fundamental data for the current time.

Ai_rules.txt uploaded at the start of the conversation:

This project is to use a sequential neural network on a dataset containing S&P1500 stocks and snapshot market volume, and close price from 4 years ago, 2 years ago, 1 year ago, 6 months ago, 3 months ago, and 1 month ago. I have the fundamental stock data for the current date.

Instructions:

- act as a senior python engineer

- use concise, production-quality python code

- when suggesting changes to code first attempt to use existing code before generating new code

- you are allowed to suggest changes to existing code in order to make new functions work better

- always explain the reasoning before showing code

- when code may be complex, add inline comments to explain code

- do more explaining in the prompt response than in code comments

- never use any emojis in responses including code comments

- prioritize pandas when making changes to the existing data

- follow PEP8 standards for generated code

- do not suggest any module for neural networks outside of tensorflow and do not suggest more complex models

- always refer back to this document before answering.

These are all of the column headers in the CSV I will be using for analysis.

address1 city state zip country phone website industry industryKey industryDisp sector sectorKey sectorDisp longBusinessSummary fullTimeEmployees companyOfficers auditRisk boardRisk compensationRisk shareHolderRightsRisk overallRisk governanceEpochDate compensationAsOfEpochDate irWebsite executiveTeam maxAge priceHint previousClose open dayLow dayHigh regularMarketPreviousClose regularMarketOpen regularMarketDayLow regularMarketDayHigh dividendRate dividendYield exDividendDate payoutRatio fiveYearAvgDividendYield beta trailingPE forwardPE volumeregularMarketVolumeaverageVolume averageVolume10days averageDailyVolume10Day bid ask bidSize askSize marketCap fiftyTwoWeekLow fiftyTwoWeekHigh allTimeHigh allTimeLow priceToSalesTrailing12Months fiftyDayAverage twoHundredDayAverage trailingAnnualDividendRate trailingAnnualDividendYield currency tradeable enterpriseValue profitMargins floatShares sharesOutstanding sharesShort sharesShortPriorMonth sharesShortPreviousMonthDate dateShortInterest sharesPercentSharesOut heldPercentInsiders heldPercentInstitutions shortRatio shortPercentOfFloat impliedSharesOutstanding bookValue priceToBook lastFiscalYearEnd nextFiscalYearEnd mostRecentQuarter earningsQuarterlyGrowth netIncomeToCommon trailingEps forwardEps lastSplitFactor lastSplitDate enterpriseToRevenue enterpriseToEbitda 52WeekChange SandP52WeekChange lastDividendValue lastDividendDate quoteType currentPrice targetHighPrice targetLowPrice targetMeanPrice targetMedianPrice recommendationMeanrecommendationKey numberOfAnalystOpinions totalCash totalCashPerShare ebitda totalDebt quickRatio currentRatio totalRevenue debtToEquity revenuePerShare returnOnAssets returnOnEquity grossProfits freeCashflow operatingCashflow earningsGrowth revenueGrowth grossMargins ebitdaMargins operatingMargins financialCurrency symbollanguage region typeDisp quoteSourceName triggerable customPriceAlertConfidence marketState earningsTimestamp earningsTimestampStart earningsTimestampEnd earningsCallTimestampStart earningsCallTimestampEnd isEarningsDateEstimate epsTrailingTwelveMonths epsForward epsCurrentYear priceEpsCurrentYear fiftyDayAverageChange fiftyDayAverageChangePercent twoHundredDayAverageChange twoHundredDayAverageChangePercent sourceInterval exchangeDataDelayedBy averageAnalystRatingcryptoTradeable shortName longName hasPrePostMarketData firstTradeDateMilliseconds postMarketChangePercent

postMarketPrice postMarketChange regularMarketChange
regularMarketDayRange fullExchangeName averageDailyVolume3Month
fiftyTwoWeekLowChange fiftyTwoWeekLowChangePercent fiftyTwoWeekRange
fiftyTwoWeekHighChange fiftyTwoWeekHighChangePercent
fiftyTwoWeekChangePercent dividendDate regularMarketChangePercent
corporateActions postMarketTime regularMarketTime exchange
messageBoardId exchangeTimezoneName exchangeTimezoneShortName
gmtOffSetMilliseconds market esgPopulated regularMarketPrice
trailingPegRatio address2 displayName fax ipoExpectedDate
prevName nameChangeDate industrySymbol prevTicker
tickerChangeDate 4y_date 4y_open 4y_high 4y_low
4y_close 4y_volume 4y_vwap 2y_date 2y_open 2y_high
2y_low 2y_close 2y_volume 2y_vwap 1y_date 1y_open
1y_high 1y_low 1y_close 1y_volume 1y_vwap 6m_date
6m_open 6m_high 6m_low 6m_close 6m_volume 6m_vwap
3m_date 3m_open 3m_high 3m_low 3m_close 3m_volume
3m_vwap 1m_date 1m_open 1m_high 1m_low 1m_close
1m_volume 1m_vwap massive_error